# **CODING:**

import os

import pandas as pd

import spacy

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

# Load spaCy English model

nlp = spacy.load("en\_core\_web\_sm")

# Step 1: Preprocessing function

def preprocess\_text(text):

doc = nlp(str(text))

return " ".join([token.lemma\_ for token in doc if not token.is\_stop and not token.is\_punct])

# Step 2: Load and prepare dataset

df = pd.read\_csv("/home/UpdatedResumeDataSet.csv") # Adjust path as per your system

# Print column names to debug

print("Columns in dataset:", df.columns)

# Rename columns (adjust based on actual column names in the dataset)

df.rename(columns={"Resume": "resume", "Category": "label"}, inplace=True)

# Binary encode label: target 'Data Science' vs others

df['label'] = df['label'].apply(lambda x: 1 if x == 'Data Science' else 0)

# Apply preprocessing to resumes

df['resume'] = df['resume'].apply(preprocess\_text)

# Step 3: Vectorize resumes

tfidf = TfidfVectorizer(max\_features=5000)

X = tfidf.fit\_transform(df['resume'])

y = df['label']

# Step 4: Split into training and testing

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Step 5: Train the model

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

# Step 6: Evaluate the model

y\_pred = model.predict(X\_test)

print(f"\nModel Accuracy: {accuracy\_score(y\_test, y\_pred) \* 100:.2f}%")

# Step 7: Screen uploaded resume

def screen\_resume(text):

clean = preprocess\_text(text) # Preprocess the input text

vector = tfidf.transform([clean]) # Vectorize the preprocessed text

pred = model.predict(vector)[0] # Predict using the trained model

return "Accepted" if pred == 1 else "Rejected"

# Step 8: Upload and test multiple resumes

def list\_and\_select\_resumes(directory):

# List all files in the directory

resumes = [f for f in os.listdir(directory) if f.endswith(".txt")]

if not resumes:

print("No resumes found in the directory.")

return []

print("\nAvailable Resumes:")

for i, resume in enumerate(resumes, 1):

print(f"{i}. {resume}")

# Allow the user to select multiple resumes by their numbers

try:

choices = input("\nEnter the numbers of the resumes you want to screen (comma-separated): ")

selected\_indices = [int(choice.strip()) for choice in choices.split(",")]

selected\_files = [os.path.join(directory, resumes[i - 1]) for i in selected\_indices if 1 <= i <= len(resumes)]

return selected\_files

except ValueError:

print("Invalid input. Please enter numbers separated by commas.")

return []

# Main execution

resume\_directory = "/home" # Directory containing resumes

selected\_resumes = list\_and\_select\_resumes(resume\_directory)

if selected\_resumes:

accepted\_resumes = []

rejected\_resumes = []

for resume\_path in selected\_resumes:

try:

with open(resume\_path, "r") as file:

uploaded\_resume = file.read()

if not uploaded\_resume.strip():

print(f"Error: The resume file '{os.path.basename(resume\_path)}' is empty. Skipping.")

else:

result = screen\_resume(uploaded\_resume)

if result == "Accepted":

accepted\_resumes.append(os.path.basename(resume\_path))

else:

rejected\_resumes.append(os.path.basename(resume\_path))

except FileNotFoundError:

print(f"Error: The file '{os.path.basename(resume\_path)}' was not found. Skipping.")

# Display results

print("\nScreening Results:")

print("\nAccepted Resumes:")

for resume in accepted\_resumes:

print(f"- {resume}")

print("\nRejected Resumes:")

for resume in rejected\_resumes:

print(f"- {resume}")